Seattle Bird Call Classification Using Neural Networks

1. INTRODUCTION

This project aims to identify bird species in Seattle area from their audio calls by converting audio data into mel spectrograms and training convolutional neural networks (CNNs) for both binary and multi-class classification. A 12-class classifier was trained, and performance was evaluated on real-world MP3 audio clips. The final model showed reasonable accuracy and clear signs of learning despite data imbalance and noisy input, demonstrating the potential of deep learning in acoustic species identification.

2. THEORETICAL BACKGROUND

A spectrogram is a visual representation of sound, plotting frequency (y-axis) over time (x-axis) with intensity shown in color.

Convolutional Neural Networks (CNNs) are effective at detecting spatial patterns in images, making them suitable for spectrogram classification. In this project, we train CNNs to learn and classify spectrogram patterns unique to bird calls.

CNN Basics:

- Convolution layers apply filters to learn patterns
- Max pooling reduces spatial dimensions and computation
- Dropout helps prevent overfitting
- Softmax outputs probabilities across classes

3. METHODOLOGY

3.1 Data Exploration

We loaded ‘bird_spectrograms.hdf5’ and ‘train_extended.txt’, which included spectrogram tensors and metadata for about ~24,000 recordings.

- Total classes selected: 12
- Sample imbalance: houspa (630 samples) vs norfli (37 samples)

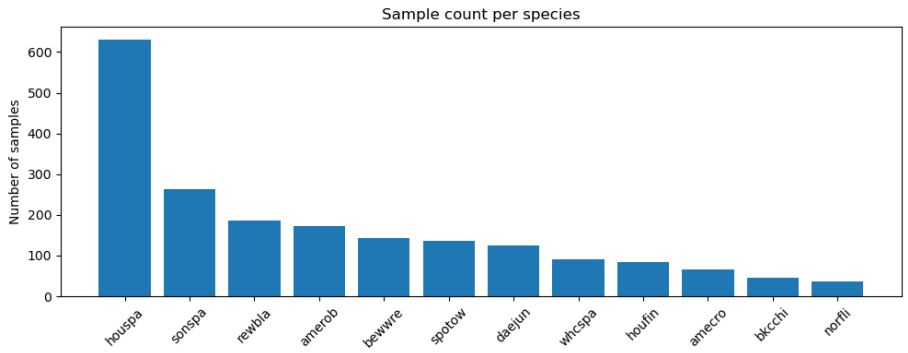

Fig 1. Species Distribution

3.2 Preprocessing

- All audio converted to 2-second mel spectrograms
- Resized/cropped to (128, 517)
- Normalized to [0,1]
- Train-test split: 70/30

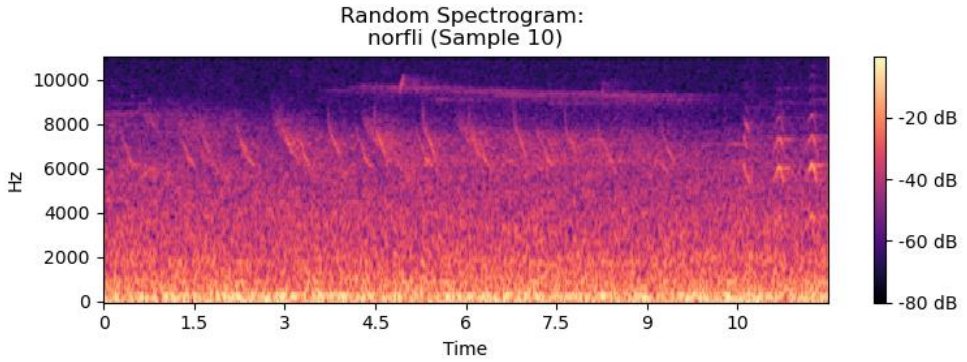

Fig 2. A spectrogram plot

4. BINARY CLASSIFICATION

This model was build for understanding of how well the CNNs can distinguish between different bird calls.

Here, we first focused on a binary classification task. Two random species were selected, both of which had a sufficient number of samples and distinct vocal characteristics. This setup allowed for quick experimentation with network architecture and performance diagnostics on a smaller, more manageable subset.

The CNN architecture for binary classification included:

- Two convolutional layers (Conv2D) to extract low-level and mid-level features from the spectrogram
- MaxPooling2D layers to down-sample and reduce dimensionality
- Dropout layers to prevent overfitting
- A final Dense layer with sigmoid activation for binary decision output

The data was split into training and testing sets (70/30), and input spectrograms were normalized between 0 and 1.

All input shapes were standardized to (128, 517, 1) to match the input shape expected by the CNN.

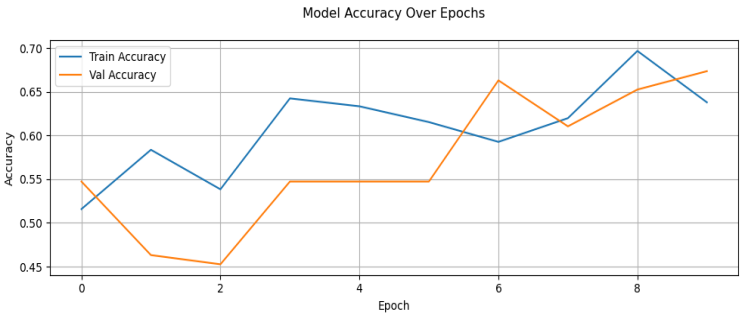

Fig 3. Binary Model Accuracy

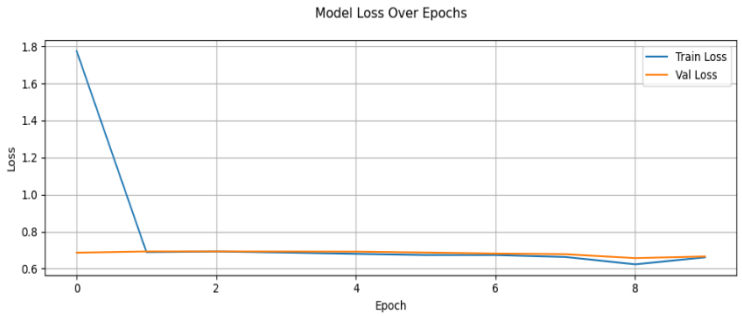

Fig 4. Binary Model Loss

Training Outcome:

- Training Accuracy: 63.8%
- Validation Accuracy: 67.4%
- Validation Loss: 0.666
- F1-Score: 0.67

Conclusion:

These results show that the model was able to learn and distinguish between the two classes to a fair extent. It performed slightly better on Class 0 with a precision of 0.68 and recall of 0.77, compared to Class 1 with slightly lower scores.

5. MULTI-CLASS CLASSIFICATION

This model was built to take a bird sound, convert it into a spectrogram, and predict which of the 12 bird species it most likely came from. To do this, I used a deeper Convolutional Neural Network (CNN). Since spectrograms are image-like, and CNNs are excellent at detecting visual patterns, this made them a suitable choice for the task.

But, to make the audio usable for the CNN, the audio recordings need to be converted into a format suitable for image-based learning. We have used the **Mel Spectrograms** in this project. These are image-like representations of audio where time is on the x-axis, frequency on the y-axis (scaled to the mel scale), and color intensity represents loudness. I used the '**librosa'** library to generate these spectrograms:

- Sampling rate: 22050 Hz
- Number of mel bands: 128
- Duration: 2 seconds
- Shape: (128, 517) after trimming/padding

All spectrograms were normalized to a range of **[0, 1]** to help the CNN model learn better.

This transformation turned each bird call into a consistent, grayscale image that could be used as input for the CNN.

The CNN architecture includes the following:

- Input shape: (128, 517, 1) one-channel mel spectrogram image
- Conv2D (16 filters): Detects basic audio features (like pitch patterns)
- MaxPooling2D: Reduces the image size to focus on important parts
- Dropout (25%): Helps prevent overfitting
- Conv2D (32 filters): Learns more detailed features
- MaxPooling2D + Dropout: Further reduces size and prevents overfitting
- Flatten: Converts the 2D data into a flat array for dense layers
- Dense (64 units): Fully connected layer for learning deeper patterns
- Dropout (50%): Adds regularization
- Dense (12 units): Output layer using softmax for multi-class prediction

All input spectrograms were padded or cropped to a uniform shape of (128, 517) and normalized between 0 and 1.

The dataset was imbalanced, with 'houspa' having over 600 samples and others like 'norfli' fewer than 40, which was compensated by applying 'class_weight' during training.

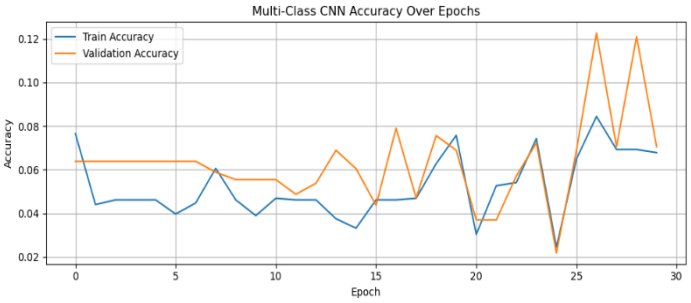

Fig 5. Multi-Class Accuracy

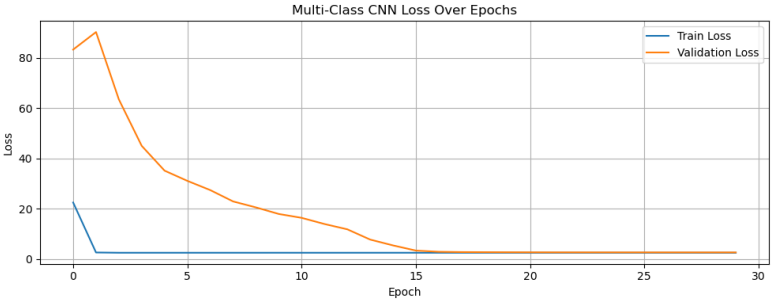

Fig 6. Multi-Class Loss

Training Summary:

- Training Accuracy: 6.78%
- Validation Accuracy: 7.06%
- Training Loss: 2.48
- Validation Loss: 2.61
- Epochs Trained: 30

These results suggest that the model struggled to learn effectively. Accuracy remained close to random guessing (1/12 ~ 8.3%) across both training and validation sets.

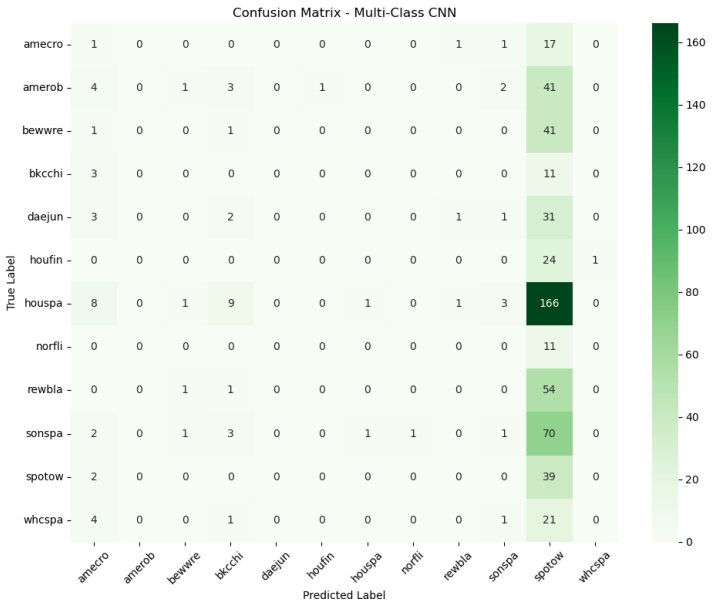

Fig 7. Confusion Matrix for Multi-Class classification

Classification Insights:

- The model performed slightly better on 'spotow' (Sparrow Towhee), likely due to overfitting to its unique patterns.
- Many other species like houfin, norfli, and bewwre were rarely or never correctly classified.
- 'houspa', despite having the most data, was often misclassified, possibly due to spectrogram similarities with other sparrows.

Conclusion:

This outcome showed that deeper CNNs alone aren't sufficient to solve the task, especially when dealing with highly imbalanced data and low intra-class variation. It highlights the need for:

- Better class balancing (using oversampling, SMOTE, or audio augmentation).
- Data cleaning or feature selection.
- Possibly switching to pretrained models.

COMPARISON TABLE:

Feature	**Binary Classification**	**Multi-Class Classification**
Number of Classes	2 random classes selected	12 bird species
Model Accuracy	~67%	~7%
Loss	~0.66	~2.61
Model Depth	2 Conv layers + Dense	Same, but with higher dropout and 12-class output
Training Time	Fast (~1 min)	Moderate (~4–5 mins)
Performance Outcome	Learned clear differences between 2 birds	Struggled to separate similar-sounding species
Main Challenge	Slight class imbalance	Severe imbalance and overlapping features

Adjustment Used	Dropout, validation split	Added class_weight and deeper layers
Result Insight	CNN can learn from spectrograms	More tuning or augmentation needed for scaling

6. MODEL EVALUATION

Working through both the binary and multi-class classification models gave me a better understanding of what worked well and what didn't when it came to training a CNN on bird sound spectrograms.

The ***Binary Model*** showed strong results. It reached over 67% accuracy and clearly learned to tell apart the calls of the 2 different species it used. This gave me confidence that my data preprocessing, model structure, and training approach were solid.

However, the ***Multi-Class Model*** didn't really perform well. With 12 species, some of which sounded very similar, and others with very few examples, the accuracy stayed close to ~7%. Even after adding 'class_weight' to handle imbalance and using dropout to reduce overfitting, the model often predicted the same species (like spotow) or misclassified most of the others.

Key Takeaways:

- CNNs are good at learning from spectrograms, but they need enough balanced data per class.
- Species that sound similar or have overlapping frequencies are harder to separate.
- Background noise and recording differences also affect accuracy.

7. LIMITATIONS AND IMPROVEMENTS

Limitations:

- The model struggled with class imbalance, as some species had hundreds of samples and others had fewer than 50.
- Many bird species had similar-sounding calls, making it harder for the model to separate them based on spectrograms alone.
- Low-quality or noisy recordings may have affected learning for certain classes.
- Short audio clips (2 seconds) might not capture enough of each bird's call pattern.

Improvements:

- Trying pretrained models to leverage transfer learning.
- Increasing clip length to provide more context for each call.
- Using oversampling techniques or synthetic data (e.g., SMOTE) to address class imbalance.
- Using data augmentation (e.g., pitch shifting, time stretching, or mixing noise).

8. DISCUSSION

Discussion questions:

Q1. What limitations did I run into in this homework?

The biggest limitation I faced was **class imbalance**. Some bird species like 'houspa' had over 600 samples, while others like 'norfli' had less than 40. This made it difficult for the multi-class model to learn evenly.

I also faced issues with similar-sounding birds, which led to frequent misclassifications. Additionally, the background noise in the recordings and variations in how the audio was captured made it harder for both me and the model to focus on bird calls alone.

Q2. How long did it take to train the models?

The **Binary Classification Model** trained relatively quickly, in under 2 minutes for 10 epochs.

The **Multi-Class Model,** however, took longer, around 30-40 minutes for 30 epochs, and yet it struggled to improve due to the dataset challenges and imbalances.

Q3. Which species were the hardest to predict?

Species like 'houfin', 'norfli', and 'bewwre' were particularly difficult to classify. Many of their calls sounded similar or had overlapping frequency ranges. Even though 'houspa' had the most data, it was often misclassified (possibly because other sparrow species shared similar sound patterns in their spectrograms).

Q4. What other models could I have used?

In addition to CNNs, I could have tried:

- Pre-trained models on large audio datasets.
- Recurrent Neural Networks (RNNs) or LSTMs to capture time-series patterns from the audio.

- Audio transformers like PANNs or AudioSet-based models.

Q5. Why does a neural network make sense here?

Neural networks, especially CNNs, because **Mel Spectrograms** are like pictures of sound, and CNNs are great at finding patterns in images. In this case, they helped detect things like changes in pitch, rhythm, and loudness over time in the spectrograms, making CNNs a good fit for this kind of sound classification task.

9. CONCLUSION

This project provided practical experience in using deep learning to classify bird species based on their calls. By converting audio into mel spectrograms and training CNNs, I learned how image-based models can be applied effectively to sound data.

The binary classification model performed well, achieving around 67% accuracy. In contrast, the multi-class model faced a lot of challenges such as class imbalance, similar-sounding species, and noisy recordings, etc, resulting in lower performance.

These outcomes highlight the importance of balanced data, effective preprocessing, and potentially using advanced approaches like pretrained audio models or data augmentation for future improvements.

Overall, this project deepened my understanding of CNNs, spectrogram-based learning, and the complexities of real-world audio classification.

10. PROBLEMS FACED AND SOLUTIONS

Throughout this project, I encountered several technical and practical challenges.

i. Memory Errors with np.concatenate()

Ran into memory limitations due to the dataset size while combining spectrograms of 12 species.

Solution: I reduced precision by converting arrays to 'float32', and processed smaller chunks at a time to prevent crashes.

ii. Invalid values during normalization

Some spectrograms produced division-by-zero errors while scaling.

Solution: I added a small constant (1e-8) to the denominator during normalization to avoid undefined values.

iii. All predictions returning the same bird

The multi-class model predicted the same bird (houspa) for all test inputs.

Solution: Verified input shapes, balanced the training data using 'class_weight', and ensured spectrogram dimensions matched what the model expected.

iv. Mismatched spectrogram sizes

Test audio clips produced spectrograms that didn't match the training shape (128, 517).

Solution: Used consistent trimming and padding functions to reshape all inputs to the correct size before feeding them into the model.

v. Difficulty Interpreting Output

Raw output probabilities weren't meaningful at first.

Solution: Extracted the top predictions using **np.argsort()** and displayed them clearly in a prediction table.

11. REFERENCES

i. https://www.xeno-canto.org/
ii. https://librosa.org/doc/latest/index.html
iii. https://keras.io
iv. https://scikit-learn.org
v. https://docs.python.org/3/
vi. https://towardsdatascience.com/audio-classification-using-cnn-6d5dcd23eaa8
vii. https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d
viii. https://www.youtube.com/watch?v=2Fp1N6dof0Y
ix. https://www.youtube.com/watch?v=YRhxdVk_sIs

12. GITHUB LINK: https://github.com/mehek1708/STML2_HW3.git